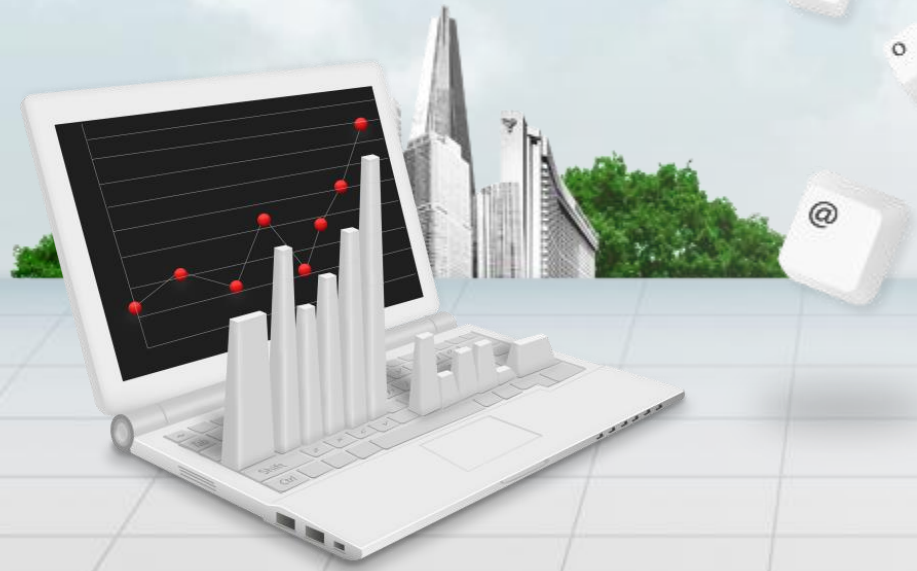
프로그래밍언어론 개요

컴퓨터공학과

이만호



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

학습목표

프로그래밍언어 이론을 구체적으로 학습하는데 필요한
기초적이고도 전반적인 내용을 학습한다.

프로그래밍언어론 개요

학습내용

- 프로그래밍언어론을 공부하는 이유
- Programming Domains
- 프로그래밍언어의 Category
- 언어설계에 영향을 주는 요인
- 프로그래밍언어의 구현 방법
- Preprocessor
- Programming 환경

목 차

알고가기

1. 다음 중에서 프로그래밍언어에 대해서 맞게 기술하고 있는 것은?

(a) 고급언어로 작성된 프로그램은 반드시 기계어로 번역된 후에 실행할 수 있다.

(b) 과학계산용 프로그래밍언어로는 기업경영을 위한 응용 프로그램을 작성할 수 없다.

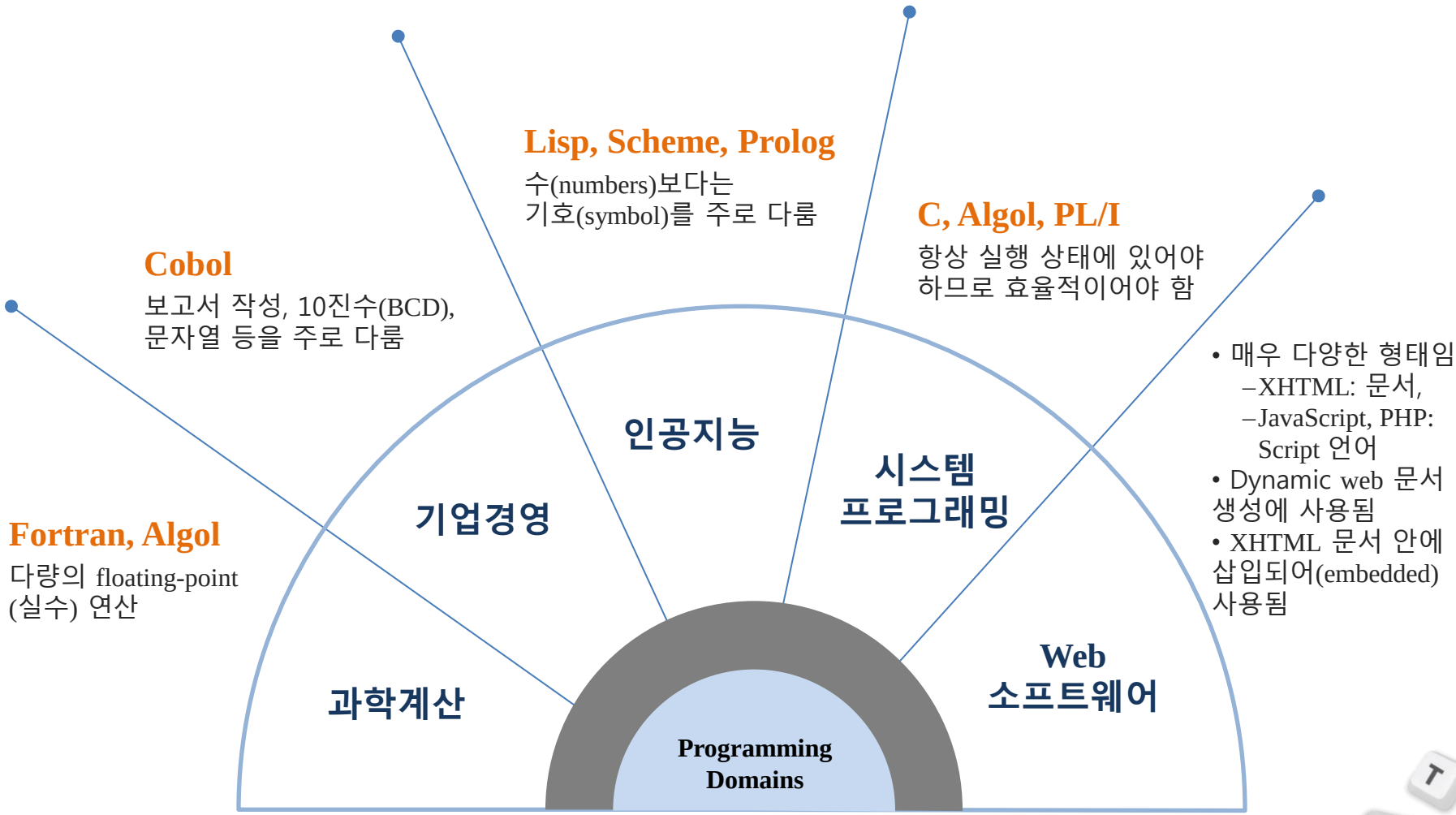
(c) 프로그래밍언어론을 공부한 후에는 새로운 프로그래밍언어를 쉽게 배울 수 있다.

(d) 컴퓨터 구조는 프로그래밍언어 설계에 별 영향을 주지 않는다.

프로그래밍 언어론을 공부하는 이유는?

- **아이디어의 표현능력 향상**
  - 표현 능력은 언어 능력에 따라 좌우됨
  - 프로그래밍 능력은 프로그래밍언어에 대한 이해에 좌우됨
- **문제에 따라 적절한 프로그래밍언어를 선택할 수 있는 능력 향상**
  - 목적에 따라 다양한 프로그래밍언어가 있음
- **새로운 프로그래밍언어를 쉽게 배울 수 있는 능력 향상**
  - 프로그래밍언어 개념을 바탕으로 한다면...
  - 새로운 프로그래밍언어가 지속적으로 출현함
- **구현의 중요성에 대한 인식 향상**
  - 프로그래밍언어는 compiler/interpreter 등으로 구현되어야 함
  - 구현 방법을 이해하면 효과적인 프로그래밍 가능
- **이미 알고 있는 프로그래밍언어의 사용능력 향상**
  - 일반적인 프로그래머는 프로그래밍언어의 일부 기능만 사용함
- **컴퓨터 역사에 대한 전반적인 이해**
  - 기능적으로 우수한 프로그래밍언어가 널리 사용되었는가?
  - 널리 사용된 프로그래밍언어는 왜 널리 사용되었는가?

Programming Domains



Ⅰ 프로그래밍언어의 Category(부류)

명령형(Imperative) 언어	- 주로 변수(variable), 배정문(assignment statement), 반복문(iteration) 등을 이용하여 프로그램을 작성함 - 예: C, Pascal, Fortran, Cobol, Algol, Basic, … - 객체지향형 언어: C++, Java, Smalltalk, … - Script 언어: JavaScript, Perl, PHP, … - Visual 언어: Visual Basic
함수형(Functional) 언어	- Parameter(매개변수)를 받아 실행하는 함수(function)를 이용하여 프로그램을 작성함 -예: LISP, Scheme, ML, Haskell, Erlang, …
논리형(Logic) 언어	- 순서는 중요하지 않은 규칙을 기반으로 함 (Rule-based) - 예: Prolog

Ⅰ 언어 설계에 영향을 주는 요인

컴퓨터의 구조

- von Neumann 구조

> 명령형 언어에 적합

프로그래밍 방법론

- 시대 흐름에 따라 변화함

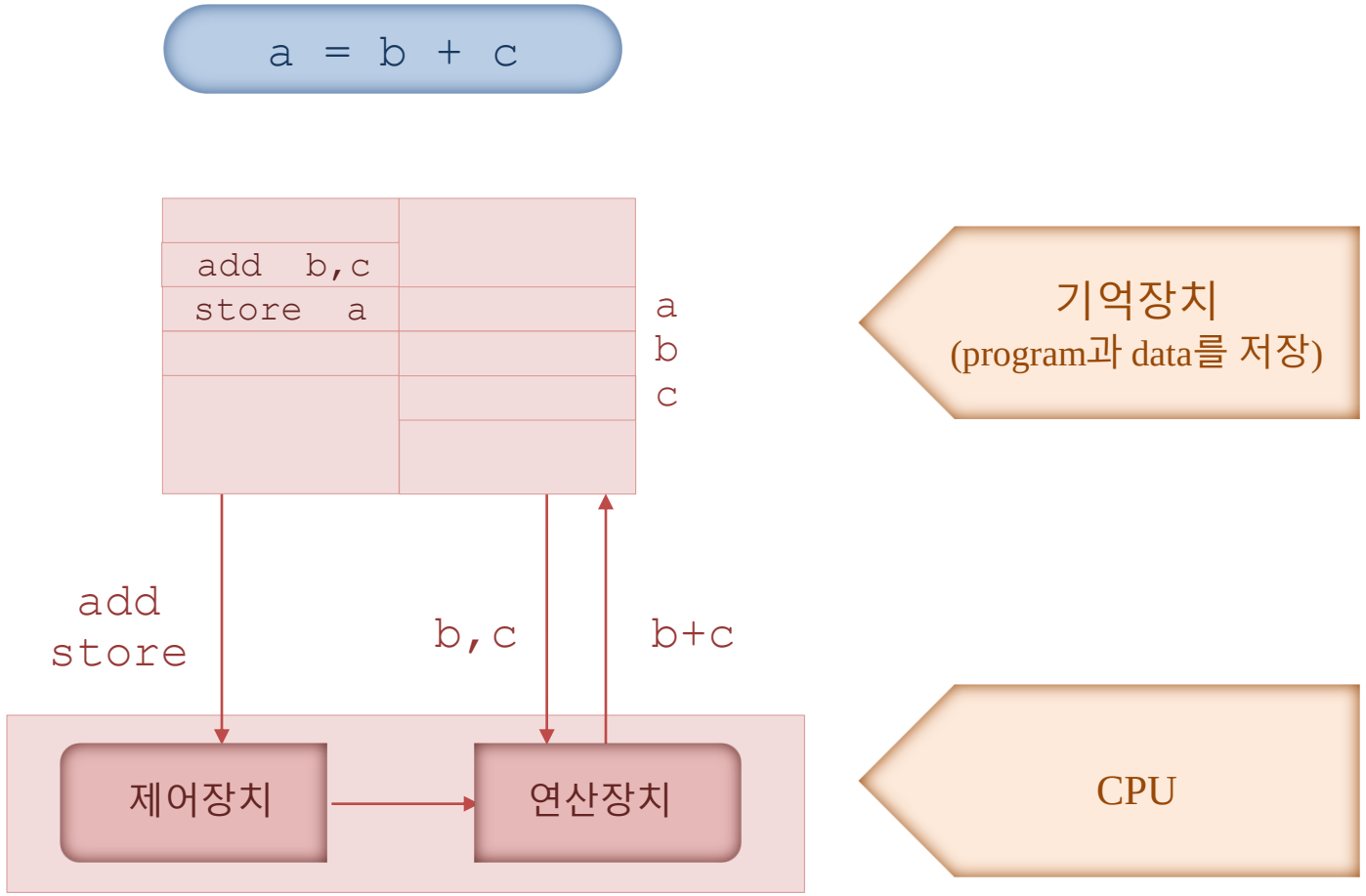
> 프로세스 지향(Process-oriented)

> 데이터 지향(Data-oriented)

> 객체 지향(Object-oriented)

- 기존 언어에 새로운 기능이 추가되어 확장되어 옴

The von Neumann 구조



The von Neumann 구조의 특성

von Neumann 구조의 특성

기억장치와 CPU가 분리되어 있음

기억장치에 프로그램과 데이터가 저장됨
> 프로그램의 명령은 제어장치로 보내어 짐 (fetch cycle)
> 연산에 필요한 데이터는 연산장치로 보내어 짐 (execution cycle)
> 연산결과는 연산장치에서 기억장치로 보내어 짐 (execution cycle)

명령형 언어가 기반을 두고 있는 컴퓨터 구조임

명령형 언어와 von Neumann 구조 사이의 관계

변수는 기억장치의 기억 소자에 해당함

배정문은 기억장치와 CPU 사이에서 명령과 데이터의 이동이 연속적으로 발생함으로써 실행됨

반복문의 명령들이 기억장치의 이웃 장소에 저장되어 있으므로 반복 실행이 효과적임

von Neumann 구조의 병목현상

병목현상(bottleneck)

프로그램이 실행되려면 **모든 명령어와 데이터가 CPU와 기억장치 사이에서 이동**해야 함

- CPU와 기억장치 사이의 데이터 이동 속도는 CPU의 처리 속도보다 매우 느림

> 기억장치 속도는 컴퓨터 처리 속도에 영향을 미치는 주요 요소임

해결 방안

- Cache 사용
- Interleaving
- Pipeline

| 프로그래밍 방법론의 변화

- 1960대 초기 이전
  - 과학계산 중심으로 컴퓨터 이용 방법이 단순함
  - 컴퓨터의 효율을 가장 중요시 함
- 1970년 전후
  - 컴퓨터 Hardware 가격 하락, 프로그래머 인건비 상승
  - 효과적으로 프로그램을 개발하는 것을 중요시 함
  -프로그램의 가독성, 효과적인 제어구조 등을 필요로 함
    > 구조적 프로그래밍(structured programming)
    > 하향식 설계(top-down design)
    > 단계적 세분화(step-wise refinement)
- 1970년대 후반
  - 추상 데이터타입을 사용해 데이터 지향 설계를 강조함
  -프로세스 지향 설계에서 데이터 지향 설계로 변화
    > 데이터 추상화(data abstraction)
  - SIMULA67
- 1980년대 초반
  - 객체지향 프로그래밍의 등장:
  -데이터 추상화 및 여러 기능들을 캡슐화
    > 프로세스 + 데이터객체 + 데이터 접근 제어 + 상속 + 동적 method 바인딩
  - Smalltalk

Ⅰ 프로그래밍언어의 구현 방법

Compiler

- 프로그램을 기계어로 번역함

  기계어로 번역된 프로그램을 기억장치에 저장한 후 실행함

순수(pure) Interpreter

- 프로그램을 문장 단위로 해석하며 순서대로 실행함

혼합형(hybrid) Interpreter

- Compiler와 순수 Interpreter가 혼합된 형태임
- 실행 방법
  - > 프로그램을 해석하기가 용이한 중간언어로 번역함
  - > 중간언어로 번역된 프로그램을 중간언어의 명령 단위로 해석하며 순서대로 실행함

Ⅰ 프로그래밍언어의 구현 방법

Compiler

- 고급언어로 작성된 프로그램을 기계어로 번역함
- 컴파일 단계
  - > 어휘 분석
  - > 구문 분석 : AST(추상구문트리) 생성
  - > 의미 분석 : 중간코드 생성
  - > 코드 생성
- 생성된 코드는 기계어로서 기억장치에 저장된 후 실행될 수 있음
- 번역 과정은 복잡하고 느림
- 실행 과정은 매우 빠름

Ⅰ 프로그래밍언어의 구현 방법

순수 Interpreter

- 프로그램을 번역하지 않음
- 실행 방법
  - > 프로그램을 문장 단위로 해석하며 순서대로 실행함
- Compiler보다 구현하기가 용이함
- Debugging이 용이함
  - > 프로그램의 source 수준의 오류 message
- 실행 속도가 매우 느림
- 실행하기 위해 많은 기억 공간이 필요함
- 예: Lisp, APL, Basic, …
- 대부분의 script 언어에 적용됨 (JavaScript, PHP 등)

Ⅰ 프로그래밍언어의 구현 방법

혼합형 Interpreter

- Compiler와 순수 Interpreter가 혼합된 형태임
- 실행 방법
  - > 어휘 분석
  - > 구문 분석 : AST(추상구문트리) 생성
  - > 의미 분석 : 중간코드 생성
  - > 생성된 중간코드를 명령 단위로 해석하며 순서대로 실행함
- 순수 Interpreter보다 실행 속도가 빠름
- Java
  - > 중간언어 : byte code
- Perl
  - ➢중간언어 : syntax tree
  - ➢ > Compile phase에서 실행하기도 하고, Run phase에서 compile하기도 함

Just-in-Time 구현 시스템

- **실행 단계**
  - 프로그램을 중간언어로 번역함
  - 부프로그램이 호출될 때
    - > 중간언어를 기계어로 compile한 후 실행함
    - > 일단 기계어로 compile된 부프로그램은 재 compile 없이 호출 가능함

- **Java와 .NET에서 사용됨**

Preprocessor (전처리기)

- Source program을 compile하기 전에 실행되는 program
- Preprocessor 명령어는 source program 안에 명시함
- Preprocessor 명령어의 용도

  - Source program에 다른 file을 삽입할 수 있음

    ```
    > #include  "myLib.h"
    ```

  - 수식을 간단히 표현할 수 있음

    ```
    > #define  max(A,B) = ((A) > (B) ? (A) : (B))
    ```

- Preprocessor 명령어를 macro 라고 부르기도 함

Programming 환경

Software를 개발할 때 사용될 수 있는 도구들의 모음
> file system, text 편집기, compiler, linker,
> debugger, 통합 도구, …

UNIX
- GUI를 이용한 환경 제공
  > Solaris CDE, GNOME, KDE

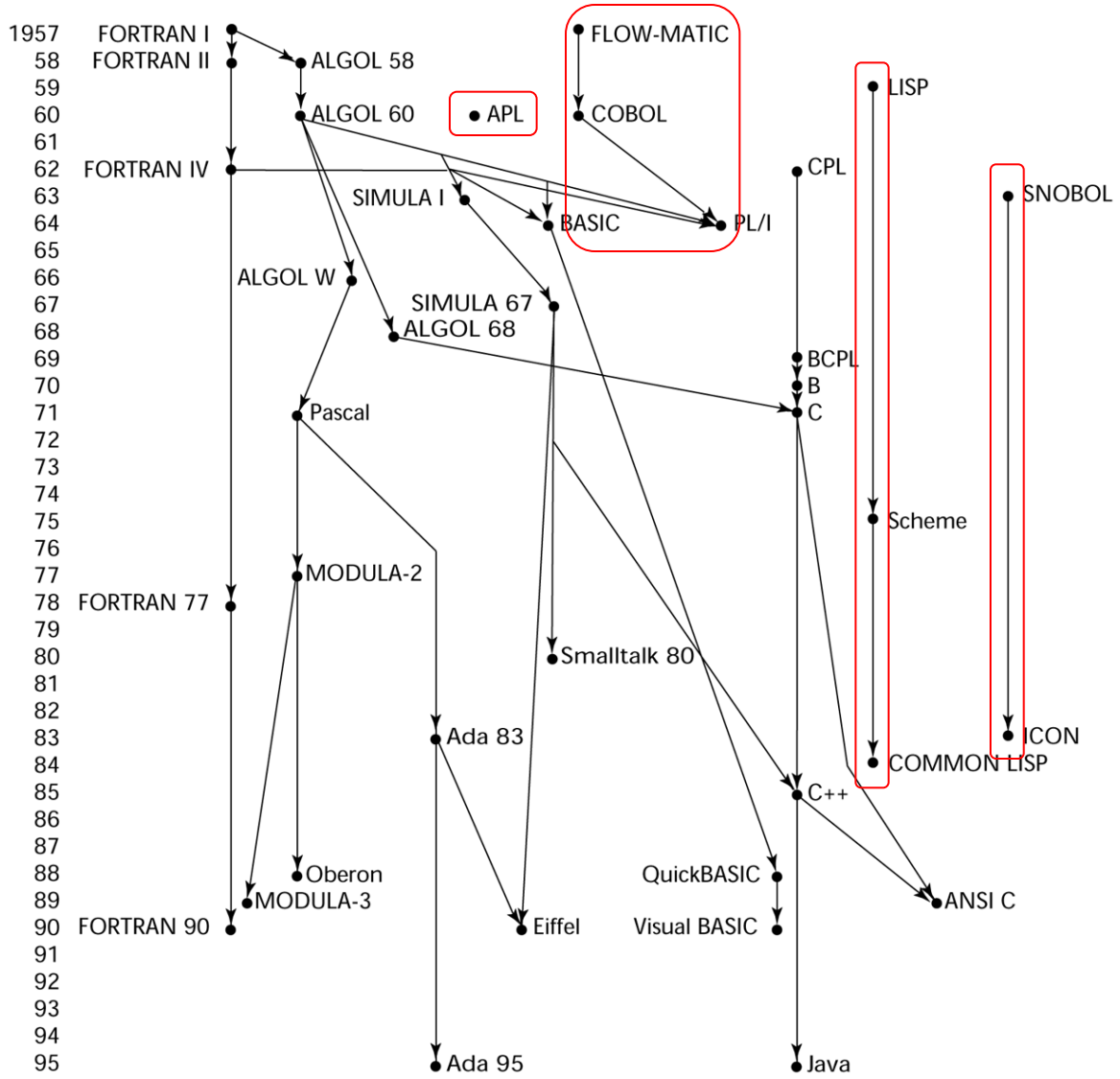
Programing 환경

Borland Jbuilder
- Java를 위한 통합 환경

NetBeans
- Web 응용 program 개발에 사용됨
- Java, JavaScript, Ruby, PHP 등을 위한 환경 제공
- Framework

Microsoft Visual Studio .NET
- Window interface를 이용한 환경 제공
- C#, Visual BASIC .NET, Jscript, J#, C++ 를 위한 통합 환경
- Framework이기도 함: 응용 program의 핵심 code를 제공함

I 프로그래밍언어의 계보



평가하기 1

1. 다음 중에서 프로그래밍언어론을 공부하는 이유에 해당하는 것은?

(a) 모든 반복 실행 프로그램을 while문으로 작성할 수 있으려고

(b) 모든 프로그램을 C언어로 작성할 수 있으려고

(c) 새로 나올 프로그래밍언어를 배우지 않으려고

(d) C 언어가 지원하는 여러 기능의 정확한 활용법을 알려고

평가하기 2

2. 프로그램 작성에 프로그래밍언어가 아닌 것을 사용할 수 있는 언어의 category(부류)는 무엇인가?

(a) 명령형(imperative) 언어

(b) 함수형(functional) 언어

(c) 논리형(logic) 언어

(d) Web 소프트웨어

평가하기 2

3. 다음 프로그래밍언어 중에서 명령형(imperative)언어가 아닌 것은?

(a) Algol

(b) Lisp

(c) Java

(d) Visual Basic

정리하기

프로그래밍언어론을 공부하는 이유

- 아이디어의 표현능력 향상
- 문제에 따라 적절한 프로그래밍언어를 선택할 수 있는 능력 향상
- 새로운 프로그래밍언어를 쉽게 배울 수 있는 능력 향상
- 구현의 중요성에 대한 인식 향상
- 이미 알고 있는 프로그래밍언어의 사용능력 향상
- 컴퓨터 역사에 대한 전반적인 이해

Programming Domains

- 과학계산, 기업경영, 인공지능, 시스템 프로그래밍, Web 소프트웨어

프로그래밍언어의 부류

- 명령형(Imperative) 언어, 함수형(Functional) 언어, 논리형(Logic) 언어

언어설계에 영향을 주는 요인

- 컴퓨터의 구조, 프로그래밍 방법론

프로그래밍언어의 구현 방법

- Compiler, 순수(pure) Interpreter, 혼합형(hybrid) Interpreter

Preprocessor

Programming 환경